A Monitoring system for Community-Lab

Navaneeth Rameshan
Universitat Politécnica de Catalunya - BarcelonaTECH
Barcelona, Spain
rameshan@ac.upc.edu

Leandro Navarro
Universitat Politécnica de Catalunya - BarcelonaTECH
Barcelona, Spain
leandro@ac.upc.edu

Ioanna Tsalouchidou
Universitat Politécnica de Catalunya - BarcelonaTECH
Barcelona, Spain
ioanna@ac.upc.edu

ABSTRACT

Community-Lab is an open and distributed infrastructure that provides a testbed for researchers to carry out experiments within wireless community networks. Community networks are an emergent model of infrastructures built with off-the-shelf communication equipment that aims to satisfy a community's demand for Internet access and ICT services. Community-Lab consists of a set of nodes integrated into the existing community networks to give researchers access to the network and to allow them to perform experiments. The challenging environment of community networks needs a careful evaluation of experimental data to understand application behavior and spot any misbehavior or anomalies. This paper focuses on demonstrating a monitoring system tailored to meet the specific requirements of the testbed and proposes an architecture for self management to automate management. This demonstration aims to present the current status of the monitoring system, the data gathered and also invite others to experiment with the data generated by the monitoring system.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

Keywords

Community-Lab; Sliver; Slice; Monitoring; Self-Management

1. INTRODUCTION

Wireless community networks are an emergent model of an infrastructure that aims to satisfy a community's demand for Internet access and ICT services. Current community networks use mainly wireless technology to interconnect nodes. Community networks pose important challenges to researchers regarding networking and applications, due to the particular conditions of how community networks are built, operated and maintained.

Management is usually decentralized with no formal service level agreements. There are no guarantees on the services offered, instead, they work on a best effort basis. The ownership is distributed and uses the commons model to share the network. Community networks are usually composed of off-the-shelf equipments and often exhibit high degree of heterogeneity both in terms of hardware and software. Since most of the links are wireless, the network exhibits dynamic behavior and the configuration of the network is constantly evolving due to different constraints like noise, limited channels available and natural obstacles.

This paper presents a demo of the monitoring system for Community-Lab [5], an experimental facility offered to researchers to conduct experiments embedded in real community networks. Community-Lab is a testbed with nodes deployed within real community networks and offers a realistic environment for the researchers to experiment with.

MobiWac'13:, November 04-08 2013, Barcelona, Spain

ACM 978-1-4503-2355-0/13/11
http://dx.doi.org/10.1145/2508222.2512838

2. COMMUNITY-LAB

Community-Lab is an open distributed research infrastructure where researchers can deploy experimental services, perform experiments or access to open data traces. Currently, Community-Lab consists of an operational testbed with more than 50 nodes deployed among three community networks in Europe. Testbed nodes are deployed in Guifi.net [3] in Spain, in Funkfeuer [2] in Austria, and in the Athens Wireless Metropolitan Network (AWMN) [4] in Greece.

A *Community-Lab node* [6] consists of two or three devices: the *community device*, the *research device* and an optional *recovery device* connected together by a wired local network, with the community device acting as a gateway. Community devices are wireless routers, while research devices are low powered PCs running a customized OpenWRT distribution that allows simultaneous running of virtual containers. Research devices are additional nodes deployed to offer applications for the community network users and extends the community network by routing traffic over them. Additionally, the research device also acts as the testbed node and provides remote access to researchers allowing them to conduct experimental studies. Each experiment uses a set of resources known as a *slice*, a set of separate virtual machines called *slivers*. Given the large scale of a community network and the experiments running on the research device, it is important to monitor the system to understand the various properties of the experiment and to spot any problems.

3. MONITORING SYSTEM

3.1 Motivation

The monitoring system is designed specifically to monitor activity on the research device. Monitoring the testbed presents specific challenges in the form of large scale of infrequently-used data. The monitoring system should support active measurements that provide insight into the functioning of nodes without revealing too much information of what is running on it, gather slice and sliver specific information and should be flexible enough to add new metrics without hampering the functionality. Monitoring logs should never lose precision and should support passively measured data such as last-time ssh succeeded, number of ports in use, resource hogs (which experiments are using the most CPU, memory, bandwidth and ports).

A general purpose monitoring system does not meet these special purpose requirements of Community-Lab and are meant for different workloads and properties. Slice specific and sliver specific information (lxc monitoring) cannot be obtained directly by any of the existing monitoring systems. Nagios, Zenoss, ntop, Ganglia, cacti [1] use RRD [8] tool for storing data. RRD is great for storing time series data and aggregating information, but are quite inflexible. It becomes necessary to compromise between flexibility and efficiency. Adding new metrics would require updating the database file. Once an RRD is created, it is possible to change existing values and add new data sources, it is not possible to add or remove metrics and change their properties. If modeling of data is not considered carefully, it can lead to a number of updates as and when new slivers are created in a node. Slice specific data implies data from different nodes and would result in a dynamic list of RRD which in turn would need additional scripts to fetch, aggregate and display data. For instance in Comon [7](monitoring system of Planetlab), the data model is carefully chosen, but still old database files are deleted when the format changes. In many cases (depending on configuration) if an update is made to an RRD series but is not followed up by another update soon, the original update will be lost. This makes it less suitable for recording data such as operational metrics. There is no way to back-fill data in an RRD series and depending on the data model, a single RRD receiving data from multiple sources can be affected by this. Given the large scale varying resource consumption and the dynamic nature of Community-Lab, flexibility is a key requirement. Apart from that, sliver-centric information is not easily integrated into node-centric data provided by off-the-shelf monitoring systems. This kind of data gathering is an important motivation for developing a separate monitoring system to meet the specific needs of Community-Lab.

3.2 Design

At a high level, the monitoring system consists of a monitoring daemon running on each research device, a centralized data gathering and processing infrastructure and a display facility. The daemon running on the research device provide node-centric data including sliver specific information and monitors periodically (e.g. every sixty seconds). It accepts HTTP requests and responds with HTTP responses, to allow them to be accessed from web browsers in addition to being used with automated systems. The response is provided in JSON format to allow researchers to query and use monitored data. The daemon stores the monitored information in a file locally until the data gathering service has seen it. This ensures that no monitored information is lost during a network partition and helps researchers diagnose any problem that may have happened during this period.

While the daemons operate on research devices, the data gathering and processing operates on a properly-provisioned machine. Data is collected from the daemons using a pull model and is fetched every 5 minutes. All fetches are performed in parallel to reduce latency. Slice centric information is generated by analyzing the node centric logs and is stored in the database. Additionally information is aggregated and summaries are provided at a granularity necessary to make meaningful inference from the data. Precision of the monitored information is never lost and it supports data offloading which is then provided as an open data-set.

Monitored information is reported via a web interface that supports sorting, and shows graphs of historical data. The reporting currently covers OS-provided metrics and metrics synthesized from other sources on the node. The system reports the following OS-provided metrics: uptime, CPU utilization, memory utilization, total memory, disk size, disk space available, 1 minute load, network data sent and received. Figure 1 shows the CPU usage over time for a research device. Synthesized data includes last time the monitoring daemon on the research device was seen, open ports, ping status and slice centric information. The status of slivers along with their IP address and resource usage is shown in figure 2. The system maintains only a manageable set of metrics that help the researchers get insight of any strange behavior in a given node. To facilitate the researchers in selecting nodes to run their experiments, the web interface provides a Treemap view of all the nodes based on the historical trend (customizable) of resource usage. Figure 3 shows a Treemap view of all the monitored research devices in the testbed.

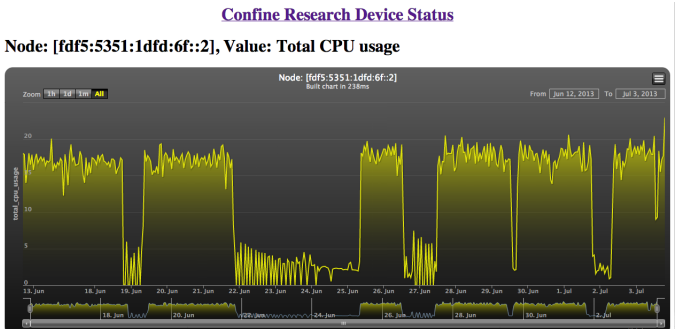

Figure 1: CPU usage history

4. SELF MANAGEMENT

The self management system should automate management at two levels: node level and slice level. As far as the node level is concerned, the system should ensure correct initial configuration of each node. The ideal behavior is well specified and the components required for correct functioning of the testbed is known beforehand. This category includes the network configuration and the configuration of

Figure 3: Treemap of Resource Usage

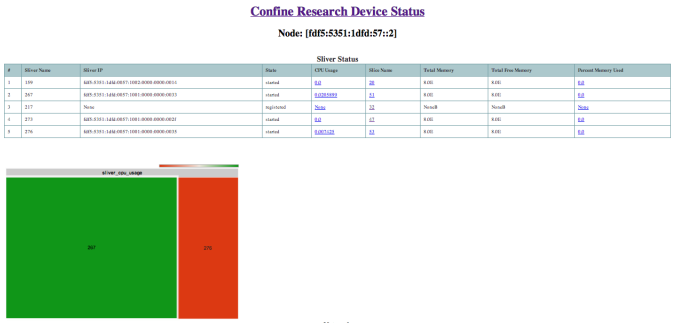

Figure 2: Current Sliver usage

the services and files. The network connection of the nodes should be correctly configured in order to ensure node visibility to the testbed controller (A central authority responsible for managing the testbed and the experiments). Moreover, the system needs to ensure that all the necessary services are properly configured and running in order to assure that the node is functional. Additionally, at the node level, the system should also ensure the physical reliability of the system, and software reliability to repair or prevent any unexpected crash of services or software. At the slice level, the system should ensure that all the slivers running under the same slice are functional and that the communication between the slivers and the slice is uninterrupted. Moreover, the slices should not exceed their limit of resource usage and should guarantee that the minimum number of slivers that should run within a slice are functional.

Figure4 shows the conceptual architecture of the self management system for the Community-Lab testbed. The system aims to close the loop on the management cycle and automate management. As shown in the figure, the monitoring daemon periodically acquires information of the disk,

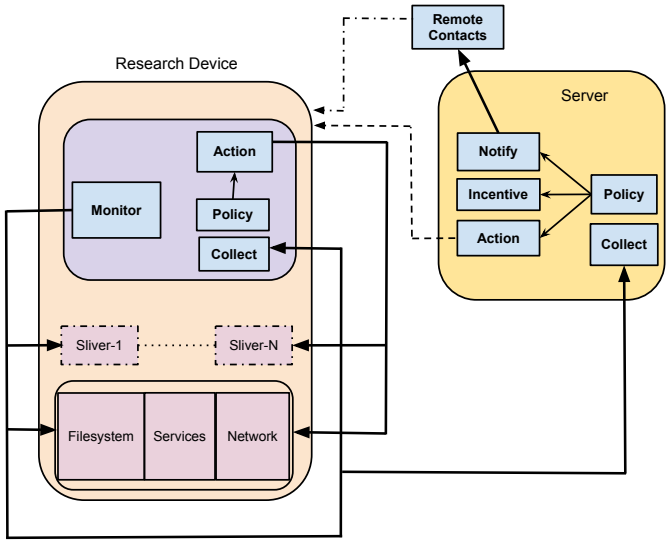

Figure 4: Self Management Architecture

CPU, services, the network and the slivers that run in each research device. The collect component in the research device gathers representational data from the monitoring daemon. A successful collection does not imply that the research device is network accessible as these components are autonomous and act locally. Using this information the system can determine the operational state of the research device. The testbed administrator specifies as policy the expected behavior, and the management actions that need to be taken upon any deviation from the expected behavior. The system detects a problem if there is any mismatch between the observed behavior from collect and the expected behavior specified in the policy. Whenever a problem needs to be addressed, the action component imposes the remedy specified in the policy. If known-to-work remedies fail to address the problem, manual intervention is required, shown as remote-contacts in figure 4. Additionally, the collect component from the server occasionally gathers data and aggregates them in order to get a view of the slices and take actions when necessary on a slice level or when a research device is unable to take local actions (ex. reboot research device, restart self management service on the research device).

5. THE DEMONSTRATION

The demonstration at the MSWiM 2013 conference is a live demo of the monitoring system for Community-Lab. The testbed which is currently composed of over 50 operational research devices will be monitored remotely.

Therefore the demonstration is composed of one computer visualizing the various monitored metrics of operational research devices. A visitor can browse through the live data and visualise the monitored metrics. It also shows slice specific data to help users easily monitor usage only corresponding to their experiments. The presentation describes how the monitored information helps a user understand the historical trend/current status of any research device, the resources consumed by experiments running on them, identifying problems, and how this information helps a researcher choose the best device for their experiment. The demo can also serve to discuss ideas for additional data that needs to be monitored which is of interest to the research community and ways of experimenting with the monitored data.

6. CONCLUSIONS

The demo of the monitoring system shows the various metrics that are monitored and how it helps to identify problems in Community-Lab, an experimental facility that allows researchers to conduct experiments in real community networks.

This demonstration aims to present the current status of the monitoring system. In a remote access to the Community-Lab testbed the demo shows the various metrics of the research devices that are monitored and also visualises them. It gives the audience a clear idea on how researchers can monitor the node and experiments along with synthesized metrics to understand the characteristics of the experiment and identify problems if any. Along with the demo, we also invite researchers to experiment with the data generated by the monitoring system.

7. REFERENCES

[1] Comparison of network monitoring tools Wikipedia.

[2] Funkfeuer (0xff), www.funkfeuer.at.

[3] Guifi www.guifi.net.

[4] Athens wireless metropolitan network (awmn), www.awmn.gr, 2010.

[5] B. Braem, C. Blondia, C. Barz, H. Rogge, F. Freitag, L. Navarro, J. Bonicioli, S. Papathanasiou, P. Escrich, R. Baig Viñas, A. L. Kaplan, A. Neumann, I. Vilata i Balaguer, B. Tatum, and M. Matson. A case for research with and on community networks. *SIGCOMM Comput. Commun. Rev.*, 43(3):68–73, July 2013.

[6] A. Neumann, I. Vilata, X. Leon, P. E. Garcia, L. Navarro, and E. Lopez. Community-lab: Architecture of a community networking testbed for the future internet. In *WiMob*, pages 620–627. IEEE Computer Society.

[7] K. Park and V. S. Pai. Comon: a mostly-scalable monitoring system for planetlab. *Operating Systems Review*, 40(1):65–74, 2006.

[8] J. Sellens. Rrdtool: Logging and graphing. In *USENIX Annual Technical Conference, General Track*. USENIX, 2006.